"Express Mail" mailing label number EV 325998005 US
Date of Deposit: September 30, 2004

Our Case No.10736/8

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Najam, et al.

Serial No. 09/858,324

Filing Date: May 15, 2001

For APPARATUS AND METHOD FOR INTERFACING WITH A HIGH SPEED BI-DIRECTIONAL NETWORK

Examiner Glenn A. Gossage

Group Art Unit No. 2187

RESPONSE AND AMENDMENT

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In reply to the Office Action dated May 6, 2004, please enter the following amendment:

In the Title:

On the cover sheet and on page 1, line 1, please change the Title as follows (the changes are shown with ~~strikethrough~~ for deleted matter and underlines for added matter):

APPARATUS AND METHOD FOR INTERFACING WITH A HIGH SPEED BI-DIRECTIONAL NETWORK USING A SHARED MEMORY TO STORE PACKET DATA

In the Drawings:

Please replace figures 3, 5, and 7 in their entirety with the replacement figures 3, 5, and 7 provided herewith.

In the Specification:

Please amend Paragraph numbers XX (as they appear in the application as filed) as follows (the changes to these paragraphs are shown with ~~strikethrough~~ for deleted matter and underlines for added matter):

[0002] U.S. patent application Ser. No. ~~_____~~ 09/858,309, "EDGE ADAPTER APPARATUS AND METHOD", (Attorney Ref. No. 10736/6), filed herewith;

[0003] U.S. patent application Ser. No. ~~_____~~ 09/858,323, "EDGE ADAPTER ARCHITECTURE APPARATUS AND METHOD", (Attorney Ref. No. 10736/7), filed herewith;

[0004] U.S. patent application Ser. No. ~~_____~~ 09/858,308, "APPARATUS AND METHOD FOR INTERCONNECTING A PROCESSOR TO CO-PROCESSORS USING SHARED MEMORY", (Attorney Ref. No. 10736/9), filed herewith.

[0006] Computer networks, in general, interconnect multiple computer systems for the purpose of sharing information and facilitating communications. Computer networks may include private networks which interconnect computers within a particular enterprise, such as an intranet, and public networks, which interconnect one or more of the computers of enterprises, public institutions and/or private individuals. One exemplary public network is the

Internet. The Internet is a packet switched network which utilizes the ~~Transport~~ Transmission Control Protocol/Internet Protocol ("TCP/IP") suite to communicate data.

[0014] The present invention is defined by the following claims, and nothing in this section should be taken as a limitation on those claims. By way of introduction, the preferred embodiments described below relate to a bi-directional data processor . The processor includes a first processor coupled with a bi-directional interface and operative to receive data from the bi-directional interface and perform a first task on the data and a shared memory coupled with the first processor, the shared memory including first and second banks. One of the first and second banks is accessible to the first processor. The first processor is further operative to store the processed data in the accessible one of the first and second banks of the shared memory. The stored processed data is then mirrored to the other of the first and second banks of the shared memory. The processor further includes a second processor coupled with the shared memory and the bi-directional interface. The second processor is operative to retrieve the stored processed data from the other of the first and second banks of the shared memory, perform a second task on the data and selectively transmit the secondarily processed data back to the bi-directional interface.

[0015] The preferred embodiments further relate to a method of processing data in a bi-directional processing device. In one embodiment, the method includes receiving the data by a first processor from a bi-directional interface, the first processor operative to perform a first task on the data, storing the processed data in a shared memory by the first processor, said shared memory comprising first and second banks, wherein one of said first and second banks is accessible to said first processor and the other of said first and second banks is accessible to a second processor, mirroring said processed data stored by said first processor in said one of said first and second banks to the other of said first and second banks, retrieving the processed data from said other of said first and second banks of the shared memory by a second processor operative to perform a second task on the processed data, thereby resulting in secondarily processed data, and transmitting, selectively, the secondarily processed data to the bi-directional interface from the second processor.

[0029] The shift to the Optical Internet has created a new set of challenges. Chief among these challenges is the need to manage an exponentially higher volume of network traffic at much higher rates of speed. In the U.S., the principal standard for optical networks is the American National Standards Institute ("ANSI") standard for synchronous data transmission over optical media known as Synchronous Optical Network ("SONET"). The SONET standard actually comprises multiple standards for transmission rates up to 9.953 gigabits per second ("Gbps") with the capability to go up to 20 Gbps. Each transmission rate standard is known as an Optical Carrier Level ("OC-X"). Exemplary optical carrier levels include OC-12 for communications at 622.08 Mbps, OC-48 for communications at 2.488 Gbps and OC-192 for communications at 10 Gbps. Today's microprocessors face a situation where they cannot support the pace of performance increases associated with the deployment of fiber-based network bandwidth of OC-48 and higher. Simply put, the move to fiber-optic networks has pushed the physical limits of microprocessors and the input/output (I/O) ~~I/O~~ bus beyond their current technical capabilities. The platform described herein is designed to address many issues associated with Optical Internet services that cannot be addressed by the current software based firewall servers.

[0030] Figure 1 shows an exemplary device 100 for intercepting and processing packets at wire speed from an optical based network 102, such as the Internet, compatible with the OC-48 standard or faster. For a more detailed explanation of the operation of devices which intercept and process packets, refer to U.S. Patent Application entitled "EDGE ADAPTER APPARATUS AND METHOD" and U.S. Patent Application Serial entitled "EDGE ADAPTER ARCHITECTURE APPARATUS AND METHOD", both of which are referenced ~~captioned~~ above. The exemplary device 100 may include the Rapid Intelligent Processing Platform manufactured by Cloudshield Technologies, Inc., located in San Jose, California. For clarity, some components of the device 100 are not shown.

[0031] The device 100 is coupled with the network 102 (consisting of an upstream network portion 102A and a downstream network portion 102B) via a network connection 110 so as to be able to intercept and process packets communicated between the upstream ~~102A~~ network portion 102A and the downstream network portion 102B of the network

102. Herein, the phrase "coupled with" is defined to mean directly connected to or indirectly connected through one or more intermediate components. Such intermediate components may include both hardware and software based components. In one embodiment, the network connection 110 is an optical network connection. In an alternate embodiment, the network connection 110 is an electrical network connection.

[0034] The device 100 further includes two secondary processing elements 112A, 112B which are coupled with the primary processing elements 104A, 104B via a command/control bus 124 and packet busses 126A, 126B, 126C, 126D. In one embodiment, each secondary processing element 112A, 112B includes a printed circuit board capable of being plugged into the backplane described above. Additional secondary processing elements 112A, 112B may be included or the functionality of the secondary processing elements 112A, 112B may be consolidated into a single secondary processing element. In one embodiment, the command/control bus 124 is a bus routed over the interconnecting backplane of device 100 and complying with the Compact Peripheral Component Interconnect ~~Personal Computer Interface~~ ("cPCI") standard and is 64 bits wide and operates at a frequency of at least 33 megaHertz (MHz) ~~MHz~~. Exemplary packet busses 126A, 126B, 126C, 126D include busses complying with the IX bus protocol of the Intel® IXP1200 Network Processing Unit, provided by Intel Corporation of Santa Clara, CA, and are described in more detail below. Each exemplary packet bus 126A, 126B, 126C, 126D may be bi-directional, 64 bits wide and operate at a frequency of at least 84 MHz and may be routed over the backplane described above. Alternatively, other bus technologies/protocols may be used and are dependent upon the implementation of the device 100. The command/control bus 124 carries command and control information between the primary and secondary processing elements 104A, 104B, 112A, 112B. The packet busses 126A, 126B, 126C, 126D carry packet data between the primary and secondary processing elements 104A, 104B, 112A, 112B.

[0035] The primary function of the secondary processing elements 112A, 112B is to perform stateful processing tasks, i.e. tasks which are dependent on historical activity. One example of a stateful processing task involves network security applications which require monitoring conversations, i.e. bi-directional packet flow, in the packet stream, typically

consisting of requests and responses to those requests. Stateful processing and the ability to monitor traffic bi-directionally allows the secondary processing elements to watch for requests and responses and match them up. The arrangement of the inbound network processors 106C of the secondary processing elements 112A, 112B, described in more detail below, allows them to share information about packets coming from either direction, i.e. upstream or downstream. Further, the secondary processing elements 112A, 112B can affect the stateless processing of the primary processing elements 104A, 104B. For example, where the secondary processing elements 112A, 112B determine that packets from a certain network address are consistently invalid, the secondary processing elements 112A, 112B can add that network address to the filtering list of the primary processing elements 104A, 104B thereby dynamically updating the stateless processing environment.

[0037] In addition, the device 100 includes a management adapter 114 which is coupled with the command/control bus 124. The management adapter 114 is used to manage the device 100 and control the functionality of the primary and secondary processing elements 104A, 104B, 112A, 112B. In one embodiment, the management adapter 114 includes a computer server having dual-PENTIUM III ~~dual-Pentium III~~ processors manufactured by Intel Corporation, located in Santa Clara, California, or suitable alternatives. The management adapter 114 further includes at least 64 MB of RAM and at least 10 GB of hard disk storage. The management adapter 114 is preferably implemented as a single board computer that plugs into the back plane, as described above, although more than one board as well as a stand alone personal computer may also be used. The management adapter 114 further includes an external management interface (not shown) which allows the connection of an external management device (not shown) for programming, controlling and maintaining the device 100. In one embodiment, the external management interface includes a model 82550 100 megabit Ethernet Interface™ manufactured by Intel Corporation, located in Santa Clara, California. Other interfaces, such as serial, parallel, coaxial and optical based interfaces may also be used. In one embodiment, the external management device is a desktop computer such as the Deskpro Model ENS SFF P733 manufactured by Compaq Computer Corporation, located in Houston, Texas. Alternatively, any suitable Pentium™ class computer having suitable memory and hard disk space in addition to Ethernet or other form of network connectivity, may be used. Further, the

external management device may be located locally with respect to the device 100 or remotely and connected to the device 100 via a local or wide area network.

[0038] The primary processing elements 104A, 104B are preferably capable of operating in parallel. The two primary processing elements 104A, 104B, are also referred to as Media Adapter Cards ("MAC") or Media Blade Adapters ("MBA"). Each primary processing element 104A, 104B includes a network interface 120, two network processors 106A, 106B, a set 122A, 122B of one or more co-processors 108, a packet bus interface 128A, 128B, and a command/control bus PCI interface 116. The network interface 120 is coupled with the network 102 via the network connection 110. In one embodiment, the network connection 110 is an optical network connection operating at a throughput of approximately 2.5 Gbps and a 1, 4 or 16 bit width. Each network processor 106A, 106B is coupled with the network interface 120, in a parallel configuration, to receive packets from the network 102. The network interface 120 converts the protocol, frequency and bus width of the network connection to the protocol, frequency and bus width of the network processors 106A, 106B. The network interface 120 further splits the incoming data among the network processors 106A, 106B, as described below. It will be appreciated that the disclosed embodiments can support any number of network processors 106A, 106B operating in parallel as described below, as the application demands. Further, each secondary processing element 112A, 112B is also coupled with network interface 120 of one of the primary processing elements 104A, 104B via the packet busses 126B, 126D to transmit packets onto the network 102, described in more detail below. The network interface 120 converts the protocol, frequency and bus width of the packet busses 126B, 126D to the protocol, frequency and bus width of the network connection 110. In addition, each network processor 106A, 106B is coupled with a set 122A, 122B of one or more co-processors 108 which is described in more detail below. Further, each network processor 106A, 106B is coupled with the command/control bus 124 via command/control interface busses 130A, 130B and the command/control bus interface 116. In one embodiment, the command/control interface busses 130A, 130B are compliant with the Peripheral Component Interconnet ~~Personal Computer Interface~~ ("PCI") standard and are 32 bits wide and operate at a frequency of at least 33 MHz. Further, the command/control bus interface 116 is a PCI to cPCI bus bridge for interfacing the busses 130A, 130B with the command/control cPCI bus 124, described above. Both network

processors 106A, 106B are also coupled with one of the secondary processing elements 112A, 112B via the packet bus interface 128A, 128B and the packet bus 126A, 126B. For a more detailed description of the primary processing element 104A, 104B, please refer to U.S. Patent Application entitled "APPARATUS AND METHOD FOR INTERCONNECTING A PROCESSOR TO CO-PROCESSORS USING SHARED MEMORY", referenced ~~captioned~~ above.

[0041] In addition, one of the network processors 106C, from each secondary processing element 112A, 112B is coupled with a set 122C of co-processors 108. It will be appreciated that the embodiments disclosed below relating to the sharing of co-processors 108 sets 122A, 122B between the two network processors 106A, 106B of the primary processing element 104A, 104B are applicable to the arrangement of the co-processors 108 and the secondary processing elements 112A, 112B. In one embodiment of the secondary processing elements 112A, 112B, the network processors 106C which are sharing the co-processors 108 of set 122C are located on two different circuit boards (one for each element 112A, 112B) which share a common daughter card containing the set 122C of co-processors 108. For more information on the arrangement and operation of the daughter cards and co-processor sets 122C, refer to U.S. Patent Application entitled "APPARATUS AND METHOD FOR INTERCONNECTING A PROCESSOR TO CO-PROCESSORS USING SHARED MEMORY", referenced ~~captioned~~ above.

[0043] Further each secondary processing element 112A, 112B handles a different direction of packet flow from the network 102. In particular, the upstream secondary processing element 112A handles packets flowing from the network ~~102A~~ upstream portion 102A of the device 100 to the network ~~102B~~ downstream portion 102B of the device 100. The downstream secondary processing element 112B handles packets flowing from the network 102B downstream of the device 100 to the network 102A upstream of the device 100.

[0044] The device 100 intercepts and processes packets from the network 102. One "upstream" primary processing element 104A intercepts packets arriving from the network ~~102A~~ upstream portion 102A of the device 100 and the other "downstream" primary processing

element 104B intercepts packets arriving from the network ~~102B~~ downstream portion 102B of the device 100. The intercepted packets are pre-processed, as described above, and then passed on to a corresponding secondary processing element 112A, 112B for subsequent processing and possible release back to the network 102. Further, within each primary processing element 104A, 104B, the network interface 120 converts the protocol, frequency and bus width of the network connection 110 to the protocol, frequency and ~~an~~ bus width of the network processors 106A, 106B and splits the incoming packet stream among the two network processors 106A, 106B which process packets in parallel (explained in more detail below). In one embodiment, the packet stream is alternated between the network processors 106A, 106B in a "ping-pong" fashion, i.e. a first packet going to one network processor 106A, 106B, the second packet going to the other network processor 106A, 106B and the next packet going back to the first network processor 106A, 106B, and so on. For more detail on this parallel packet processing architecture, refer to U.S. Patent Application entitled "EDGE ADAPTER ARCHITECTURE APPARATUS AND METHOD", referenced ~~captioned~~ above. The network processors 106A, 106B are further coupled with the packet bus interface 128A, 128B which couples both network processors 106A, 106B with the common packet bus 126A, 126C to the secondary processing elements 112A, 112B.

[0046] For packets traveling from the network ~~102B~~ downstream portion 102B of the device 100 to the network ~~102A~~ upstream portion 102A of the device 100, the packets are intercepted by the network interface 120 of the downstream primary processing element 104B. The network interface 120 passes the intercepted packet to one of the network processors 106A, 106B which preliminarily process the packet as described above. This may involve the shared co-processors 108, as described below. The packet is then transmitted to the inbound network processor 106C of the downstream secondary processing element 112B for subsequent processing via the packet bus interface 128B and packet bus 126C. Within the downstream secondary processing element 112B, the packet is processed and moved from the inbound network processor 106C to the outbound network processor 106D via the SDRAM memory fabric 118. This processing may involve processing by the shared co-processors 122. If it is determined that the packet is to be released, in original or modified form, the outbound network processor 106D sends the packet to the network interface 120 of the upstream primary

processing element 104A via the packet bus 126D. The network interface 120 of the upstream primary processing element 104A then transmits the packet back onto the network 102A.

[0048] The network processor 106A, 106B, 106C, 106D used in the primary and secondary processing elements 104A, 104B, 112A, 112B is preferably a general purpose network processor which is suitable for a wide variety of network applications. In one embodiment, each primary and secondary processing element 104A, 104B, 112A, 112B includes two network processors 106A, 106B, 106C, 106D and supporting hardware (not shown), as described above. An exemplary network processor 106A, 106B, 106C, 106D is the Intel® IXP1200 Network Processor Unit, manufactured by Intel Corporation, located in Santa Clara, California. For more detailed information about the exemplary processor 106, please refer to Intel® IXP1200 Network Processor Datasheet part no. 278298-007 published by Intel Corporation, located in Santa Clara, California. This exemplary network processor 106A, 106B provides six micro-engines/path-processors for performing processing tasks and a StrongARM™ control processor. Each of the network processors 106A, 106B, 106C, 106D preferably operates a frequency of 233 MHz or faster, although slower clock speeds may be used. It will be appreciated that other network specific or general purpose processors may be used.

[0049] Figure 2 shows a detailed diagram of the secondary processing element 112A, 112B. The secondary processing element 112A, 112B, includes one inbound network processor ~~processors~~ 106C and one outbound network processor 106D. The inbound network processor 106C is coupled with the inbound packet bus 126A, 126C from the primary processing elements 104A, 104B. The outbound network processor 106D is coupled with the outbound packet bus 126B, 126D to the primary processing elements 104A, 104B, as described above. Both network processors 106C, 106D are coupled with command/control bus interfaces 116A, 116B via the command/control interface busses 130C, 130D which interface the network processors 106C, 106D to the command/control bus 124. In one embodiment, the command/control interface busses are 32 bit PCI compliant busses operating at 33 MHz and the commands/control bus interfaces 116A, 116B comprise PCI to cPCI bridge interface 116A, 116B which interfaces the network processor 106A, 106B to the 64 bit wide 66 MHz command/control cPCI bus 124. The command/control cPCI bus 124 interconnects the primary

processing element 104A, 104B with other processing elements 104 and other resources (not shown) of the device 100 allowing the sharing of data and distribution of packet processing. The PCI to cPCI bridge interface 116A, 116B includes a model I21154, manufactured by Intel Corporation, located in Santa Clara, California. Alternatively, other bus architectures may be used for communication among the components of the device 100.

[0051] Each network processor 106C, 106D, is further coupled with a bank 210A, 210B of synchronous ~~sync~~ burst static random access memory ("SSRAM") via an SSRAM data interface 212A, 212B and an SSRAM address interface 214A, 214B provided by the network processor 106C, 106D. In one embodiment, the SSRAM data and address interfaces 212A, 212B, 214A, 214B are each 32 bits wide and operate at a frequency of at least 100 MHz. Each bank 210A, 210B of SSRAM includes a block 218A, 218B ~~216A, 216B~~ of one or more single port SSRAM devices.

[0053] The inbound network processor 106C is further coupled with SRAM control logic 228 via the SRAM control logic address interface 230. The SRAM control logic 228 ~~228A, 228B~~ is also coupled with the DPSSRAM block 216 via the control logic DPSSRAM interface 240. The DPSSRAM block ~~blocks~~ 216 is also coupled with the daughter card 122C via the DPSSRAM daughter card interface 254A. In one embodiment, the DPSSRAM daughter card interface is at least 32 bits wide and operates at a frequency of at least 100 MHz. The SRAM control logic 228 is coupled with the daughter card 122C ~~204~~ via SRAM control logic daughter card interfaces 252A. In one embodiment, the SRAM control logic 228 is a custom designed device using a CMOS Programmable Logic Device ("CPLD").

[0054] The secondary processing element 112A, 112B further includes a daughter card connector 246 for connecting an auxiliary printed circuit board, also known as a daughter card, 122C to the main circuit board 202 of the secondary processing element 112A, 112B. The elements 112A, 112B are interleaved such that the Net processor 106C, 106D access 112A, 112B as 32 bit devices at the dictated nominal clock of 100 MHz. The elements 112A, 112B interface through connector 246 to co-processor control logic 236 with a 64 bit bus. This implementation allows the same throughput from the Net processor but at half the clock speed

to the daughter card and the co-processor control logic ~~interface~~ 236. In one embodiment, the daughter card connector 246 includes a 140 pin high density connector. An exemplary high density connector is the QStrip™ QTE/QSE series connector from Samtec, Inc. located in New Albany, Indiana. Alternatively, other connector interfaces and protocols may be used. An exemplary configuration for the connector 246 is (MB = main circuit board 202, CC = daughter card 122C):

SYMBOL	Direction	DESCRIPTION
GND	MB to CC	Ground
RST#	MB to CC	Chip reset.
MCLK	MB to CC	Chip Master clock.
DP_A(1)	CC to MB	Dual-Port SRAM address.
DP_A(2)	CC to MB	Dual-Port SRAM address.
DP_A(3)	CC to MB	Dual-Port SRAM address.
DP_A(4)	CC to MB	Dual-Port SRAM address.
DP_A(5)	CC to MB	Dual-Port SRAM address.
DP_A(6)	CC to MB	Dual-Port SRAM address.
DP_A(7)	CC to MB	Dual-Port SRAM address.
DP_A(8)	CC to MB	Dual-Port SRAM address.
DP_A(9)	CC to MB	Dual-Port SRAM address.
DP_A(10)	CC to MB	Dual-Port SRAM address.
DP_A(11)	CC to MB	Dual-Port SRAM address.
DP_A(12)	CC to MB	Dual-Port SRAM address.
DP_A(13)	CC to MB	Dual-Port SRAM address.
DP_A(14)	CC to MB	Dual-Port SRAM address.
DP_WE#	CC to MB	Dual-Port SRAM write enable.
DP_CE#	CC to MB	Dual-Port SRAM chip enable.
DP_CLK	CC to MB	Dual-Port SRAM clock.
DP_D(63:0)	Bi-direction	Dual-Port SRAM data.
AFC_D(63:0)	MB to CC	Address Filter Chip data.
AFC_RD#	CC to MB	Address Filter Chip read enable.

AFC_CLK	CC to MB	Address Filter Chip read clock.
AFC_FFSEL	CC to MB	Address Filter Chip FIFO select.0: CAM FIFO1: Classify FIFO
AFC_FF#	MB to CC	Address Filter Chip Full Flag.
AFC_EF#	MB to CC	Address Filter Chip Empty Flag.
TCK	MB to CC	Boundary-scan Test-Access-Port clock.
TMS	MB to CC	Boundary-scan Test-Access-Port mode select.
TDI	MB to CC	Boundary-scan Test-Access-Port input data.
TDO	MB to CC	Boundary-scan Test-Access-Port output data.
CPGM#	MB to CC	Configuration Programming.
CINIT#	MB to CC	Configuration Init.
CCLK	MB to CC	Configuration Clock.
CDIN	MB to CC	Configuration data input to CC FPGA.
CDONE	CC to MB	Configuration done.

[0055] The daughter card 122C includes daughter card control logic 236 and a set of co-processors 108. In one embodiment, the co-processors 108 includes two classification co-processors and eight content addressable memories ("CAM") cascaded to supply CAMs ~~CAM~~ and classification banks. The daughter card control logic 236 interfaces with the DPSSRAM daughter card interface 254A, 254B and the SRAM control logic daughter card interface 252A, 252B of each secondary processing element 112A, 112B. The daughter card control logic 236 is further coupled with each of the co-processors 108 via co-processor interfaces 238. Each co-processor may further be coupled with each other in a cascaded fashion via an ~~a~~ inter-co-processor interface bus (not shown). It will be appreciated that other components located on the main circuit board 202 can be moved to the daughter card 204 and vice versa depending on the implementation of the processing element 104 and the desired performance requirements.

[0056] In one embodiment, both inbound network processors 106C of each secondary processing element 112A, 112B share the same set ~~122C~~ of co-processors 108. The daughter card control logic 236 interfaces all of the co-processors 108 to both inbound network processors 106C. Alternatively, each processing element 112A, 112B may have its own set

122C of co-processors 108. For example, each secondary processing element 112A, 112B may include its own daughter card 122C. For more detail on the operation of the SRAM control logic 228, the DPSSRAM 216 and the daughter card 122C, refer to U.S. Patent Application entitled "APPARATUS AND METHOD FOR INTERCONNECTING A PROCESSOR TO CO-PROCESSORS USING SHARED MEMORY", referenced ~~captioned~~ above.

[0057] Both network processors 106C, 106D are further coupled with an SDRAM memory fabric 118 via SDRAM interfaces 222A, 222B. In one embodiment, the SDRAM interfaces 222A, 222B are 64 bits wide and operate at a frequency of at least 100 MHz. The SDRAM memory fabric 118 includes memory banks 220A, 220B consisting of synchronous dynamic random access memory ("SDRAM") for use as working/code storage and inter-processor communications. It will be appreciated that other types of memory may also be used, such as asynchronous dynamic RAM or static RAM. Further, the SDRAM control logic 224 is also coupled with the inbound network processor's 106C SDRAM interface 222B via SDRAM control logic interface 226 and to the outbound network processor's 106D SDRAM bank 220A via a memory interface 232 which allows shared operation between the network processors 106C, 106D, described in more detail below.

[0058] Figure 3 shows a more detailed diagram of the SDRAM memory fabric 118. The SDRAM memory fabric 118 includes an SDRAM memory 220 which is logically divided, as seen by these network processors 106C, 106D, into a portion 220A for use by the inbound network processor 106C and a portion 220B for use by the outbound network processor 106D. The SDRAM memory fabric 118 is coupled with SDRAM interfaces 222B, 222A of each network processor 106C, 106D. It will be appreciated that the protocols and operation of these interfaces 222A, 222B are dependent of the type of network processor 106C, 106D used in the particular implementation. These interfaces 222A, 222B carry both data and addressing information. As will be described below, the inbound network processor's 106C SDRAM interface 222B is coupled with inbound working storage SRAM bank 308 via the inbound unshared memory bus 314 and with the inbound shared SDRAM bank 304 via the inbound shared memory bus 316, together which form the inbound SDRAM interface 222B. The outbound network processor's 106D SDRAM interface 222A is coupled with the outbound

working storage SDRAM bank 308 via the outbound unshared memory bus 328. The outbound network processor's 106D SDRAM interface bus 222A is further coupled with each replica of the outbound shared SDRAM banks 306A, 306B via the first and second outbound shared memory busses 324, 326. Together, the unshared outbound memory bus 328 and first and second outbound shared memory busses 324, 326 form the outbound SDRAM interface bus 222A.

[0061] In general, the SDRAM memory fabric 118 permits the network processors 106C, 106D to share a segment of common SDRAM memory space 332 with each network processor 106C, 106D assuming that it has full control over the entire shared memory space 332 at all times. Each network processor 106C, 106D is capable of both reading and writing to the entire SDRAM memory 220. However, the operation of the network processors 106C, 106D and the operation of the SDRAM control logic 224 restricts the inbound network processor 106C to exclusively writing data to the shared SDRAM memory area 332 and restricts the outbound network processor 106D to exclusively reading data from the shared SDRAM memory area 332. In this way, the two network processors 106C, 106D, each operating uni-directionally, together operate bi-directionally.

[0066] Typically, at any given time, outbound network processor 106D will be in control of one replicated outbound shared memory bank ~~banks~~ 306A, 306B and the SDRAM Control Logic 224 controller 310 will be in control of the other replicated outbound shared memory bank 306A, 306B. During the time between successive replicated outbound shared memory bank 306A, 306B switches, the outbound network processor 106D may issue SDRAM Active, Pre-charge or Refresh commands to the replicated outbound shared memory bank 306A, 306B that it is currently controlling. The SDRAM control logic 224 must track these commands, and duplicate them in the other replicated outbound shared memory bank 306A, 306B before the next switch may take place. This ensures that both first and second replicated outbound shared memory banks 306A, 306B are properly synchronized before the next switch is made.

[0067] The inbound network processor 106C processor and the outbound network processor 106D processor are each running on their own independent clocks. Even though they

both use the same frequency setting for their clock, and they both have an SDRAM clock frequency of 100 MHz, as described above, the clocks are asynchronous with respect to each other. As a result, one of the functions of the SDRAM control logic 224 is to provide an interface between these two asynchronous systems. The SDRAM control logic 224 uses the cache 312 and the SDRAM Control Logic 224 controller 310 as the system clock boundary. Referring ~~Refer~~ to Figure 5, careful design techniques are employed at this clock boundary to prevent meta-stability from occurring in any data, address or control signal.

[0070] The following definitions will be useful in understanding the remaining figures:

CAS	Column Address Strobe ~~Select~~;
CASL	Column Address Strobe ~~Select~~ Latency;
RAS	Row Address Strobe ~~Select~~;
MUX	Multiplexer;
ZDG	Zero Delay Gate (a ZDG is a logic gate/switch with a low " switch on" time (within the same clock cycle as the switch command, hence "zero delay") to avoid latency (delay) for data transfers. It is essentially, a very fast switch;
Bit	One "Bit" is a single binary digit;
Byte	8-bit data width;
Word	Word (16-bit / 2-byte data width);
DWord	Double Word (32-bit / 4-byte data width);
QWord	Quad Word (64-bit / 8-byte data width).

[0071] Figure 4 shows a state diagram 400 which details the flow of data and control for the SDRAM fabric 118 as implemented by the SDRAM control logic 224. In the figure, "CPU-1" refers to the inbound network processor 106C and "CPU-2" refers to the outbound network processor 106D. "FPGA" refers to the SDRAM control logic 224 controller 310. "Data FIFO" and "ADDRESS FIFO" refer ~~refers~~ to the cache 312. Two processes of the SDRAM control logic 224 are shown in figure 4, namely a CPU-1 write access packet data transfer into packet FIFOs, and packet FIFOs/FGPA packet data transfer into SDRAM "B" and

"C" for CPU-2 read access. With regard to a CPU-1 write access packet data transfer into packet FIFOs process, the SDRAM control logic 224 begins in an idle state, and remains there as long as CPU-1 does 'not WRITE'. If CPU-1 writes data, SDRAM control logic 224 evaluates the CPU-1 write. If the CPU-1 write is for "CODE DATA", the SDRAM control logic 224 returns to the idle state. If the CPU-1 write is for "PACKET DATA", the SDRAM control logic 224 moves to a store packet data (data FIFO) state. Note, that if CPU-1 is still writing in a "PACKET DATA" burst, the SDRAM control logic 224 will remain in this state. Once the CPU-1 write burst is complete, the SDRAM control logic 224 enters a store packet address (addr FIFO), wherein the packet address is stored into a packet FIFO. Once the packet address FIFO store is complete, the SDRAM control logic 224 returns to the idle state. With regard to the packet FIFOs/FGPA packet data transfer into SDRAM "B" and "C" for CPU-2 read access process, the SDRAM control logic 224 begins in a precharge SDRAM B state. If SDRAM "B" is idle, the SDRAM control logic 224 checks the packet address FIFO. If the FIFO has an entry for packet SDRAM "B", the SDRAM control logic 224 write the packet data to the SDRAM "B". Once the packet data write burst is complete, the SDRAM control logic 224 returns to check the packet address FIFO. When the FIFO has no more entries for packet SDRAM "B", the SDRAM control logic 224 refreshes SDRAM "B". When the refresh is complete, the SDRAM control logic 224 synchronizes SDRAM "B" with CPU-2. The SDRAM control logic 224 remains in this state as long as the system is not ready for a switch between SDRAM "B" and SDRAM "C". When the system is ready for the switch between SDRAM "B" and SDRAM "C", the SDRAM control logic 224 couples SDRAM "B" to CPU-2 and SDRAM "C" to the FPGA. SDRAM control logic 224 then uses similar processes to transfer packet data into SDRAM "C".

[0072] The SDRAM Control Logic 124 ~~118~~ controller 310 is a key piece of the SDRAM control logic 124 ~~118~~ that provides the data pump and control logic to keep ~~both~~ memories 304, 306A, 306B mirrored and serviced, as well as providing both network processors 106C, 106D with constant access to the shared memory 332. The controller 310 consists of two major sections split on clock boundaries. These are the inbound network processor 106C and outbound network processor 106D Data Controller sections. Refer to Figure 5. Some of the functions of the controller 310 include:

Selecting the correct data (i.e. "packet" data) from inbound network processor 106C to store in the shared memory 332;

Maintaining inbound network processor 106C packet data in the cache 312 during the mirroring process;

Maintaining inbound network processor 106C packet address in the cache 312 during the mirroring process;

Maintaining data synchronization between the two memories 304 and 306A, 306B(mirror);

Maintaining command synchronization between the ~~two~~ memories 304 and 306A, 306B;

Arbitrating the Ping-Pong switching between the ~~two~~ memories 306A, 306B;

Interfacing between two asynchronous clock domains (inbound network processor 106C & outbound network processor 106D clock domains);

Controlling the outbound network processor 106D access between packet SDRAM 306A, 306B and code SDRAM 308;

Resetting the SDRAM control logic 124 ~~118~~;

Configuring the Packet SDRAM during boot-up; and

Duplicating the outbound network processor's 106D SDRAM Active, Pre-charge and Refresh commands.

[0077] The inbound network processor 106C Bank State Machine 504 defines the control logic used to resolve the various inbound network processor 106C SDRAM bus transactions, and then to set or reset the appropriate Bank Status Register locations. Figure 6 depicts a state diagram 600 of the bank status register state machine. As shown, the system can enter any state from any other state, depending on the transaction. All Banks Status Registers are reset on either a system reset or a precharge all Banks transaction. On a default transaction, the system then goes to an idle state, wherein no operation ("NOP") is performed. On a pre-charge transaction for a selected Bank, the selected Bank Status Register is reset. If an active transaction is issued for a selected Bank, the selected Bank Status Register is set.

[0079] The inbound network processor 106C SDRAM Data Bus 222B has a 2-register pipeline 540 before the Packet Data FIFO. This pipeline 540 gives the controller 310 one clock cycle to determine if a Packet Data write is in process before it must initiate a Packet Data FIFO 312 write. Reference will be made hereinafter to both cache 312 and FIFO (or FIFO's) 312. Once a Packet Data FIFO 312 write starts, the Write Controller 506 continues to monitor the Command Decoder 502 and Bank Status Register 504 to determine if DQM (data mask) has been asserted on any cycle, and when the Write burst is complete. When the burst is complete, the Write Controller 506 halts the Packet Data FIFO 312, sends DQM information and the burst length to the Address Entry Write Register 508. The full address information is then written in the Packet Address FIFO 312.

[0080] The Packet FIFO Write Controller 506 State Machine defines the control logic used to resolve the various inbound network processor 106C SDRAM bus 222B transactions, and then move the Packet Data and Address into the appropriate FIFOs ~~FIFO's~~ 312 during a Packet Data Write. Figure 7 depicts a state diagram 700 of the packet FIFO write controller 506 state machine. On a system reset, the packet FIFO write controller 506 enters an idle state. On a 'wr & pkt bank' transaction, the packet FIFO write controller 506 enters a burst 1 state, wherein the packet FIFO write controller 506 write the Packet Data into the FIFO 312. The packet FIFO write controller 506 then continues through burst states 2 through 8, acting in a similar fashion. After reaching burst state 8, the packet FIFO write controller 506 enters a write end state, wherein the Address is stored in the appropriate FIFO. The packet FIFO write controller 506 the returns to the idle state. If a 'wr & pkt bank' transaction is present when in any of the burst states, the packet FIFO write controller 506 enters an echo burst state, wherein the Packet Data and Address information is moved into the appropriate FIFO. If a 'not wr & pkt bank' transaction is present, when in any of the burst states, the packet FIFO write controller 506 enters the write end state.

[0082] The controller 310 includes clock domain translation logic 510. The inbound network processor 106C processor and the outbound network processor 106D processor

are each running on their own independent clocks. Even though they both use the same frequency setting for their clock, and they both have an SDRAM clock frequency of 100 MHz, the clocks are asynchronous with respect to each other. As a result, one of the functions of the SDRAM control logic 124 ~~118~~ is to provide an interface between these two asynchronous systems. The SDRAM memory fabric 118 uses the Packet Data FIFO 312, the SDRAM Control Logic 124 ~~118~~ controller 310 and the Packet Address FIFO 312 as the system clock boundary. Refer to Figure 5. The design employed at this clock boundary prevents meta-stability from occurring in any data, address or control signal.

[0085] The Packet Address and Data FIFO Write Address Pointers are a bit more complex. These Address Pointers are generated in the inbound network processor 106C Data Controller 538 section, and passed from the inbound network processor 106C to the outbound network processor 106D clock domain. The Address Pointers consist of many bits of data that all must arrive in the outbound network processor 106D clock domain at the same time (i.e. on the same outbound network processor 106D clock edge). However, if re-synchronization registers are simply used (as for the Global Warm-Reset), meta-stability concerns would occasionally cause some of the Address Pointer bits to arrive on one outbound network processor 106D clock edge, while some other bits would arrive on one clock edge (or very rarely two clock edges) ~~clock edges~~ later. This may cause the system to lose synchronization.

[0087] The primary function of the outbound network processor 106D Data Controller 512 is to monitor the Packet Address FIFO 312 for queued packet data bursts, and then to write those data bursts into both first and second replicated outbound shared memory banks 306A, 306B, completing the data mirror. To achieve this goal, the outbound network processor 106D Data Controller 512 must also perform numerous sub-functions including maintaining outbound network processor 106D command synchronization and arbitrating the Ping-Pong switching between the two Packet SDRAM's 306A, 306B. The outbound network processor 106D Data Controller 512 also switches between Packet SDRAM 306A, 306B ~~306D~~ and Code SDRAM 308 during read operations, and handles system warm-reset and boot-up.

[0092] The Packet SDRAM Write Controller 522 issues its Auto Refresh commands just prior to a SDRAM switch command when all the SDRAM banks are in the Idle state. Because of this, several Auto Refresh commands may have been counted since the last SDRAM switch, and a Refresh Request Counter keeps track of the number of these commands.

[0093] Just prior to the Packet SDRAM Switch there is a several clock window where another Auto Refresh command could be received from the outbound network processor 106D, but it would be too late to be reissued to the SDRAM by the Packet Write Controller 522. In this case, the Refresh command would be memorized and issued to that same Packet SDRAM 306A, 306B the next time its control is switched back to the Packet SDRAM Write Controller 522. There are two Refresh Request counters in order to keep track of Refresh commands for both replicated outbound shared memory banks 306A, 306B ~~307B~~. Figure 8 depicts a state diagram 800 of the outbound network processor refresh request state machine 518. On a system reset, the outbound network processor refresh request state machine 518 enters a reset state, wherein the RefreshReqCntB variable and RefreshReqCntC are set to 0. By default, the system enters an idle state, wherein no operations are performed. On a refresh request, the system increments the appropriate count. On a refresh done, the system decreases the appropriate count.

[0096] The Controller 310 Packet SDRAM Write Controller 522 goes into action right after it receives confirmation of a successful replicated outbound shared memory bank 306A, 306B switch ~~has been received~~ from the Switching Logic circuit 526. Once the switch is confirmed, the Write Controller 522 performs a Pre-charge All command in order to Idle all Active SDRAM Banks. If the Packet Address FIFO 312 is not empty, the Write Controller 522 fetches the next FIFO 312 entry. If the Bank and Row Address of the fetched address are currently Active, a Write Command is issued. If the Bank and Row Address are either Idle or set to a different address, then the appropriate Pre-charge and Active Commands are issued by the Write Controller 522, followed by the Write Command.

[0097] When the Packet SDRAM Write Controller 522 issues a SDRAM Write Command, it checks the Burst Length for this write sequence. If the Burst Length is only 1 Quad

Word, then a Burst Terminate Command is issued on the next cycle. If the burst length is more than 1 Quad Word, then the Write Controller 522 continues to push out write data on each clock. When the number of Quad Words sent matches the Burst Length, but is not a full length burst (less than 8 Quad Words), then a Burst Terminate Command is issued on the next cycle. If a full-length burst (8 Quad Words) was sent, then the write sequence is complete, and the Write Controller 522 goes onto the next state. Please note that since the SDRAM control logic 124 ~~118~~ is set to use a burst size of 8, then any smaller burst sizes must be truncated by a Burst Terminate Command.

[0098] Once the write burst is complete the Write Controller 522 checks to see if there are other entries in the Packet Address FIFO 312, and if the maximum number of writes (256 bursts) has been processed. If the FIFO 312 has more entries and the write counter has not exceeded its maximum, then the next address is ~~if~~ fetched, and another write burst initiated. If the FIFO 312 is empty or 256 write bursts have been processed, then the Write Controller 522 stops performing writes, and prepares the Packet SDRAM 306A, 306B to switch back to outbound network processor 106D control.

[0099] To prepare for the SDRAM Switch, the Write Controller 522 first issues a Pre-charge All command to the SDRAM, to put all banks in an Idle state. Then the Write Controller 522 checks if any Refresh Commands have been received from outbound network processor 106D, and it issues the same number of Refresh Commands to the SDRAM. Next, the Write Controller 522 checks the outbound network processor 106D Bank Status Register 524, and sets each of the SDRAM Banks to match the Idle, Active and Row Address parameters of the Status Register 524. If the Status Register 524 changes during this process (due to an outbound network processor 106D Active or Pre-charge Command being issued), the Write Controller 522 will duplicate those changes before moving to the next step.

[00101] During system boot-up (or warm-reset) the Write Controller 522 performs some special functions. Right after boot-up, the Switching Logic 526 bridges both SDRAM's "B" & "C" 306A, 306B to the SDRAM Control Logic 124 ~~118~~ controller 310. The controller 310 has full control over these 2 SDRAM's 306A, 306B. The Write Controller 522 monitors

outbound network processor 106D SDRAM boot-up commands (Pre-charge and Refresh), and mimics those commands to the SDRAM's 306A, 306B. When the Write Controller 522 sees outbound network processor 106D issue the "Set SDRAM Mode Register" command, it modifies that command before sending it on to the SDRAM's 306A, 306B. The Code SDRAM has a CAS Latency set to 3, while the Packet SDRAM's need a CAS Latency set to 2. The Write Controller 522 makes this change and then sets the Packet SDRAM Mode Registers. After the Mode is set, two Refresh Commands are issued to the SDRAM's 306A, 306B (per the SDRAM specification), and then the Write Controller 522 commands the Switching Logic 526 to "Break the Controller 310 / SDRAM Bridge", and commence normal operations. The only way to repeat this process once normal operations have started is to issue a "Warm-Reset" command to the Controller 310 via the I2C interface. This will halt all Controller 310 operations, purge all FIFO's 312, and put the Controller 310 in a "just woke up" state.

[00103] Figure 9 depicts a state diagram 900 of the packet SDRAM write controller 522 state machine. The Write Controller 522 State Machine shown in the figure is single-threaded, performing only one command during any given state. This "single-thread" version of the state machine meets the 100 MHz system clock requirements. Following a system reset, the Write Controller 522 enters an idle state. From the idle state, three commands may be issued. Following a refresh command, the Write Controller 522 refreshes the SDRAM. Following a pre-charge command, the Write Controller 522 precharges all banks. Following a set SDRAM mode command, the Write Controller 522 sets the SDRAM mode register, refreshes the SDRAM twice, sets the switch logic to the end FPGA bridge, and finally precharges all banks. At this point and if the FIFO 312 is not empty, the Write Controller 522 will fetch the next burst address. If the fetched bank is active, but the row is idle, the Write Controller 522 precharges the bank and activates the bank and row. If the bank is idle, the Write Controller 522 only activates the bank and row. If both the bank and row are active, the Write Controller 522 writes the data in a burst. If the burst is for multiple write commands, the Write Controller 522 will continue to push write data to the SDRAM at the fetched address until the burst is done. Following the completion of the burst, the Write Controller 522 will continue to fetch addresses and write data to the SDRAM until the FIFO is empty or the bank is full. Once either of these conditions is met, the Write Controller 522 precharges all banks and refreshes the SDRAM.

Next, the Write Controller 522 continuously activates the bank and row and precharges the banks as needed until the banks are synchronized, at which point the Write Controller 522 switches between the SDRAMs.

[00108] The Switching Logic & State Machine 526 monitors the outbound network processor 106D Command Decoder 514 and the Packet SDRAM Write Controller 522 in order to determine the correct time switch control of Packet SDRAM's "B" and "C" 306A, 306B between the SDRAM Control Logic 124 ~~118~~ controller 310 and the outbound network processor 106D.

[00113] When the Switching Logic 526 is in the Check for Burst Read state, it will count 8 clock cycles to ensure that if a Burst Read is in process, it will complete before the switch is made. At the end of 8 clocks, the Switching Logic 526 will automatically make the Packet SDRAM switch. In addition, if a Read or Write Command is ~~are~~ decoded while waiting for the 8 clocks to pass, it is now a perfect time to perform the switch, and the switch will occur immediately.

[00114] The Switching Logic 526 also has a special state that is used when coming out of reset. Right after reset, outbound network processor 106D will send out the SDRAM Mode Register Set command. However, the Packet SDRAM's 306A, 306B need a setting that is slightly modified from the one used by the Code SDRAM 308. To accommodate this need, the SDRAM Control Logic 124 ~~118~~ controller 310 modifies the Code SDRAM 308 Set Mode command and issues it to the Packet SDRAM's 306A, 306B. To permit this, the Switching Logic 526 bridges both first replicated outbound shared memory bank 306A & "C" to the SDRAM Control Logic 124 ~~118~~ controller 310 while it sets the replicated outbound shared memory banks 306A, 306B Mode Registers. After the Mode Register is set, the Switching Logic 526 commences its normal operation (as described earlier in this section). Figure 10 depicts a state diagram 1000 of the packet SDRAM "B" "C" 306A, 306B switch state machine. Following a system reset, the system sets each SDRAM 306A, 306B mode and enters an idle state. If the Write Controller 522 is ready for a switch, the system will bridge the SDRAM to the SDRAM Contol Logic 124 controller 310. If a write command is issued, the system will switch

SDRAMs. Otherwise, the system will check for a random read command. If no read command is issued within the prior 3 clock cycles, the system will check for a burst read command. At this point, the system will switch SDRAMs only if a write or read command is issued, or if 8 clock cycles pass.

[00115] The outbound network processor 106D Code / Packet SDRAM Output Enable Control & State Machine 528 monitors the outbound network processor 106D Command Decoder 514, outbound network processor 106D Bank Status Register 524, and the outbound network processor 106D Bank Address in order to track Packet SDRAM ("B" or "C") 306A, 306B and outbound working storage bank 308 access cycles. The Output Enable Control 528 then determines if the Code Data ZDG (Zero Delay Gate) or the outbound network processor 106D Packet Read Data Register shall have its ~~it's~~ outputs enabled.

[00117] In addition, since an ~~a~~ SDRAM control logic 124 ~~118~~ rule (or limitation) states that outbound network processor 106D will only "read" from the Packet SDRAM 306A, 306B, this output enable controller 528 assumes that all outbound network processor 106D "write" accesses must be for the Code Data SDRAM 308. Second, if a DQM is issued by outbound network processor 106D during a Packet SDRAM burst read, the output enable controller 528 switches over to the Code Data ZDG for that cycle. This is because the DQM may be preparing for a Code Data write, and this controller must anticipate that possibility. This rule is shown in Figure 11 which depicts ~~a~~ the state diagram 1100 of the code/packet SDRAM output enable state machine. As shown, the system remains in a 'code data state' if a burst terminate, read & code bank, write, or precharge (this bank), end of burst, or a DQM command. The system will only enter the read packet data state on a read & packet bank & not DQM command.

[00118] A standard I2C™ Philips serial bus standard based on a two wire protocol slave interface 530 allows the Controller 310 Control Register to be set, and Status Register read by an external controller. More information about this serial bus standard may be found at the web site semiconductors.philips.com/i2c/, last accessed May 15, 2001. It is not required to establish communications with the Controller 310 since all the control registers boot-up with a

default value. However, it is necessary if any of the default values need to be changed. The maximum data & clock rate of the I2C interface is approximately 25 MHz.

[00119] The Packet SDRAM Control Logic 124 ~~118~~ controller 310 Interrupt Controller 532 is provided to alert the network processors 106C, 106D of a warning or error condition. Interrupt sources include a number of FIFO level warnings, FIFO overflow, and a Packet Address FIFO vs. Packet Data FIFO misalignment. Refer to the table below.

[00120] The Interrupt output may be configured to provide positive or negative logic, and may be either an edge (50 nanoseconds (nS) ~~nS~~ pulse), or level output. If level output is selected, reading the interrupt source register clears the Interrupt output.

[00122] The controller 310 status register 536 is detailed in the table below:

FUNCTION	LOGIC	I2C ADDRESS	I2C BIT POSITION
SDRAM Control Logic 124 ~~118~~ controller 310 ID	Binary value	Base address + 1	0 – 7
SDRAM Control Logic 124 ~~118~~ controller 310 Rev.	Binary value	Base address + 2	0 - 7
FIFO 50% Full Interrupt Enabled	'0' = Disabled		
'1' = Enabled	Base address + 3	0	
FIFO 75% Full Interrupt Enabled	'0' = Disabled		
'1' = Enabled	Base address + 3	1	
FIFO 90% Full Interrupt Enabled	'0' = Disabled		
'1' = Enabled	Base address + 3	2	
FIFO 100% Full Interrupt Enabled	'0' = Disabled		

'1' = Enabled	Base address + 3	3	
Address & Data FIFO misalign Int. En.	'0' = Disabled		
'1' = Enabled	Base address + 3	4	
FIFO 50% Full Interrupt source	'0' = not intr. Source		
'1' = interrupt source	Base address + 4	0	
FIFO 75% Full Interrupt source	'0' = not intr. Source		
'1' = interrupt source	Base address + 4	1	
FIFO 90% Full Interrupt source	'0' = not intr. Source		
'1' = interrupt source	Base address + 4	2	
FIFO 100% Full Interrupt source	'0' = not intr. Source		
'1' = interrupt source	Base address + 4	3	
Address & Data FIFO misalign Interrupt	'0' = not intr. Source		
'1' = interrupt source	Base address + 4	4	
FIFO 25% Full Status	'0' = FIFO < 25% full		
'1' = FIFO > 25% full	Base address + 5	5	

FIFO 50% Full Status	'0' = FIFO < 50% full		
'1' = FIFO > 50% full	Base address + 5	0	
FIFO 75% Full Status	'0' = FIFO < 75% full		
'1' = FIFO > 75% full	Base address + 5	1	
FIFO 90% Full Status	'0' = FIFO < 90% full		
'1' = FIFO > 90% full	Base address + 5	2	
FIFO 100% Full Status	'0' = FIFO < 100% full		
'1' = FIFO overflow	Base address + 5	3	
Address & Data FIFO misalign interrupt	'0' = FIFO OK		
'1' = FIFO's Misalign	Base address + 5	4	

Please amend the Abstract as follows (the changes on these paragraphs are shown with ~~strikethrough~~ for deleted matter and underlines for added matter):

~~An inter-processor communications interface is disclosed. The interface~~ A method and apparatus for processing a bi-directional dataflow are disclosed which permits the transparent movement of data from one processor to another via a shared memory fabric which is connected with both processors. This permits the incoming data of a first processor to be utilized by a second processor thereby freeing that processor from having to handle incoming data. Further, the second processor can handle outgoing data exclusively, freeing the first processor from

having to handle outgoing data. In this way, each direction of a bi-directional dataflow may be handled by the maximum capability of a bi-directional capable processing device. The shared memory may comprise a plurality of banks of synchronous dynamic random access memory (SDRAM) devices, and may be used to store packet data in a network.